ESD-TR-68-290

MTR-711

DATA FLOW IMPROVEMENTS: DESIGN SPECIFICATIONS FOR AFICCS SERIAL FILE MANIPULATORS

O. Beebe
J. Penney
J. Terrasi

NOVEMBER 1968

Prepared for

DIRECTORATE OF PLANNING AND TECHNOLOGY
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 512V
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract AF19(628)-5165

DATA FLOW IMPROVEMENTS: DESIGN SPECIFICATIONS FOR AFICCS SERIAL FILE MANIPULATORS

O. Beebe
J. Penney
J. Terrasi

NOVEMBER 1968

Prepared for

DIRECTORATE OF PLANNING AND TECHNOLOGY
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 512V

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts

Contract AF19(628)-5165

FOREWORD

Contractor: The MITRE Corporation
Bedford, Massachusetts 01730

Contract Number: AF19(628)-5165

Air Force Contract Monitor: Charles L. Bruce, ESLFA

This report defines the overall design for an AFICCS generalized serial tape file management capability. The effort was performed between 2 January 1968 and 15 April 1968.

This report was issued on 1 May 1968.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved.

WILLIAM F. HEISLER, COL, USAF
Chief, Command Systems Division
Directorate of Planning & Technology

ABSTRACT

This document defines the overall design for an AFICCS generalized serial tape file management capability. This capability provides three management functions — generate, update, and browse.

TABLE OF CONTENTS

SECTION I

1.0 INTRODUCTION

The AFICCS generalized serial tape file management capability is designed to provide three types of file manipulation functions — generate, update and browse. These functions are controlled by specifications defined via a user-oriented language.

One of the salient design features of the capability is the ability to examine and edit both input transactions and current file data. In addition to the well known file generation and update functions, the browse function will operate in conjunction with the OCC (BR-90), thus providing the necessary man-machine interface to perform on-line auditing and verification of existing file data.

Two modes of operation are available: off-line batched processing and on-line conversational employing OCC interaction. The OCC interface may be utilized with all types of file manipulations, but is mandatory with the browse function.

SECTION II

2.0 GENERAL CAPABILITY DESCRIPTION

The serial tape management package is a set of programs which can be broadly categorized according to their functions, namely, executive, translator, and utility routines. The system executive provides the overall supervision, control, and subprogram sequencing for the three types of file manipulations (generate, update, and browse) and is driven by the control set information. One set of control information (directives) contains the serial tape file description, the input data description, and the logical data operations* to be performed. It is the function of the system translator to convert the logical data operations from their external form to an internal representation for subsequent interpretation and execution by an executive subroutine. The primary purpose of the utility routines is to build and maintain:

1) tables such as conversion and validity lists;
2) subroutines such as special error checking programs; and
3) complete control set information including the translated logical operators.

For efficient operation, the executive program has been separated into two distinct phases. Phase 1 accepts input data and, based on the associated file/input descriptions and tables, generates transaction records on tape. Similarly, Phase 2 accepts the input transaction records and, based on the associated logical operators, tables, and subroutines, produces logical records which when merged with the current file results in an updated file.

The three management functions, via the executive, appeal to all or a subset of Phase 1 and Phase 2 routines. Update utilizes both Phase 1 and Phase 2 routines. Browse causes all Phase 2 routines to be executed; Phase 1 is not executed since browse does not require input data. Generate utilizes all Phase 1 and Phase 2 routines.

---

*Logical operators are defined in Appendix A.

SECTION III

3.0 DETAILED PROGRAM DESCRIPTIONS

A total of fourteen programs comprise the serial tape management package. Prior to presenting descriptions of these specific routines, definitions of the system elements must be established.

3.1 System Definitions

- file record – tape record to be generated or updated
- logical record – core image of the file record
- unit record – card/tape input record, one or more of which are used to generate/update a logical record
- control set – information defining the file and maintenance process. Each control set consists of:
  - file attribute list – description of the file structure
  - unit record input attribute list – description of the unit record types and structure
  - logical operators – user-defined processors of logical/file record data.

3.2 Executive Routines

Four routines – FMEXEC, PHASE1, PHASE2, and TEMPMOD – perform the executive functions including control card interpretation, job sequencing and input/output operations.

FMEXEC, the file management executive routine, interprets the system control cards* and, if requested, executes the Logical Operator Temporary Modifier Routine (TEMPMOD). Subprograms PHASE1 and PHASE2 are called for generate and update manipulation actions; PHASE2 is called for a browse request.

---

*Control card formats are specified in Appendix D.

PHASE1, the unit record processor, reads the input data and generates logical record transaction data under control of the control set directives.

PHASE2 merges logical record transaction data with file data, or if a browse request, scans and error detects/corrects current file data.

TEMPMOD, called by FMEXEC, allows temporary exception updates of logical operators at run time.

3.3 Translator Routine

One routine — TRANS — translates raw logical operators to an internal representation.

TRANS, called by the executive routines, accepts the logical operators in an input buffer and, using file and unit record descriptions, converts them to an internal executable representation.

3.4 Utility Routines

Eight utility processors maintain the control set information, update tables and subroutines, in addition to providing in appropriate printed format all maintenance information associated with the system files.

CSDB, the control set processor, deletes existing control set entries or creates new control set skeletons.

FALB, the file attribute list processor, adds or redefines file attribute lists for a given control set and appeals to the logical operator processor to retranslate, if required.

URALB, the unit record attribute list processor, creates or redefines the unit record attribute List for a given control set and calls for retranslation of the logical operators, if required.

LOB, the logical operator processor, translates initial logical operator sets or retranslates existing logical operator sets for given control set. The processor will, at user option, save raw operators on tape.

SUBDB, the subroutine processor, updates the subroutine directory by:

1) adding new entries;
2) changing SDA's of existing subroutines; or
3) deleting entries.

TABB, the table processor, updates tables by:

1) adding new table entries;
2) changing SDA's of existing tables; or
3) deleting entries.

ERRDB, the error directory processor, updates entries in the error directory.

UPRINT, the utility print program, lists in appropriate formats:

1) the control set dictionary;
2) the control set file attributes, unit record structure and attributes;
3) table data; and
4) the subroutine directory.

3.5 OCC Interface Routine

OCCIF, the OCC interface program, accepts error messages including the erroneous data (in stream format) from the 1410 and allows error correction, if desired, for subsequent transmission to the 1410. OCCIF is a BR-90 program.

APPENDIX A

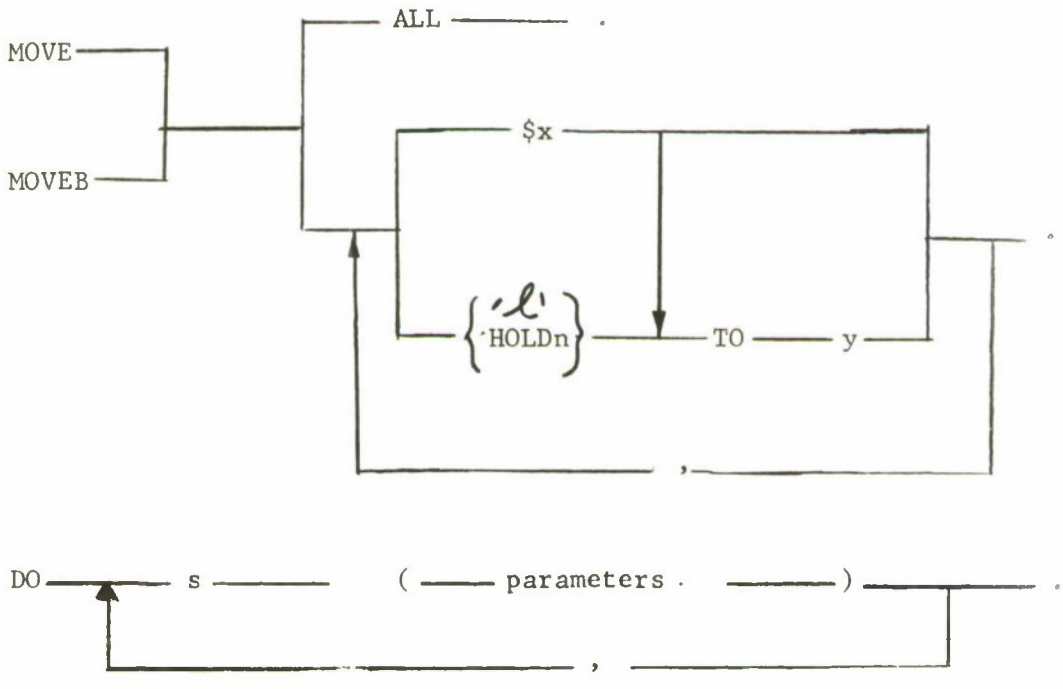
LOGICAL OPERATOR DEFINITIONS AND SYNTAX

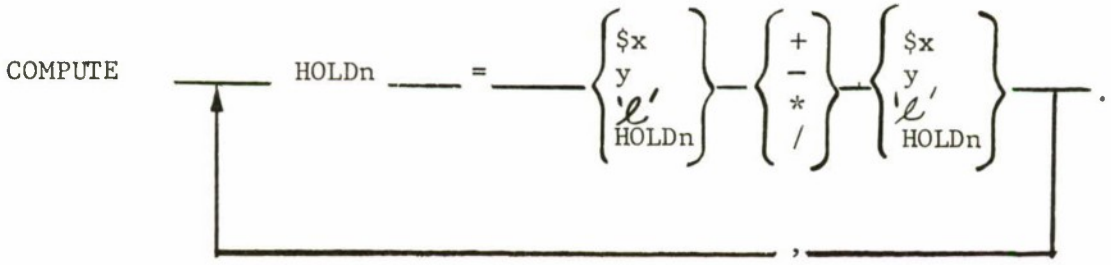
DEFINITIONS:

MOVE	Move field to output (blank fields not moved).
MOVEB	Move field to output (blank fields moved).
DO	Execute user subroutine.
COMPUTE	Compute to system hold area.
SET	Set input error.
IF	Compare function.
$x	Input field representation where x is the attribute name.
y	Output field representation where y is the attribute name.
'ℓ'	Literal representation where ℓ is the literal.
HOLDn	Hold area representation where n identifies the particular hold field.
s	Subroutine representation where s is the subroutine name.
e	Error representation where e is the error number.
Parameters	Subroutine parameters separated by slashes (/).
True phrase	Any combination of SET, MOVE, MOVEB, DO and/or COMPUTE operators.
False phrase	
_____	One or more blank characters (blanks around , or . are optional).
{ }	Choice of one entry.

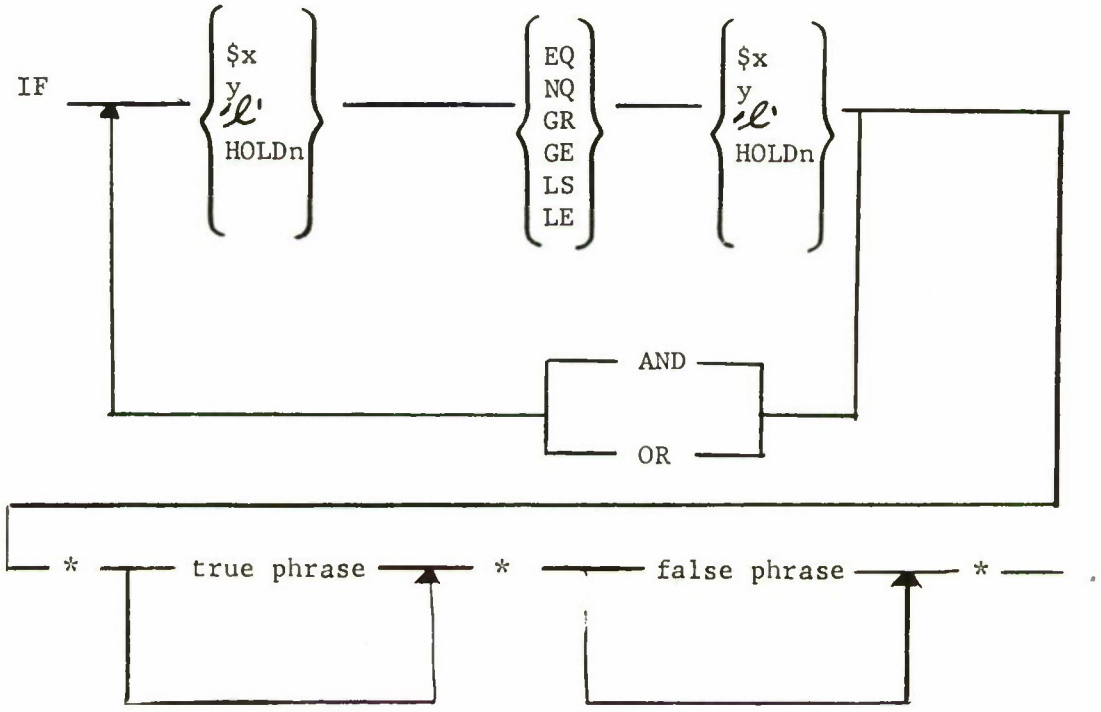
+	Add
−	Subtract
*	Multiply
/	Divide
EQ	Equals
NQ	Not equal
GR	Greater
GE	Greater or equal
LS	Less
LE	Less or equal

SYNTAX:





(Valid only in IF T/F phrase).



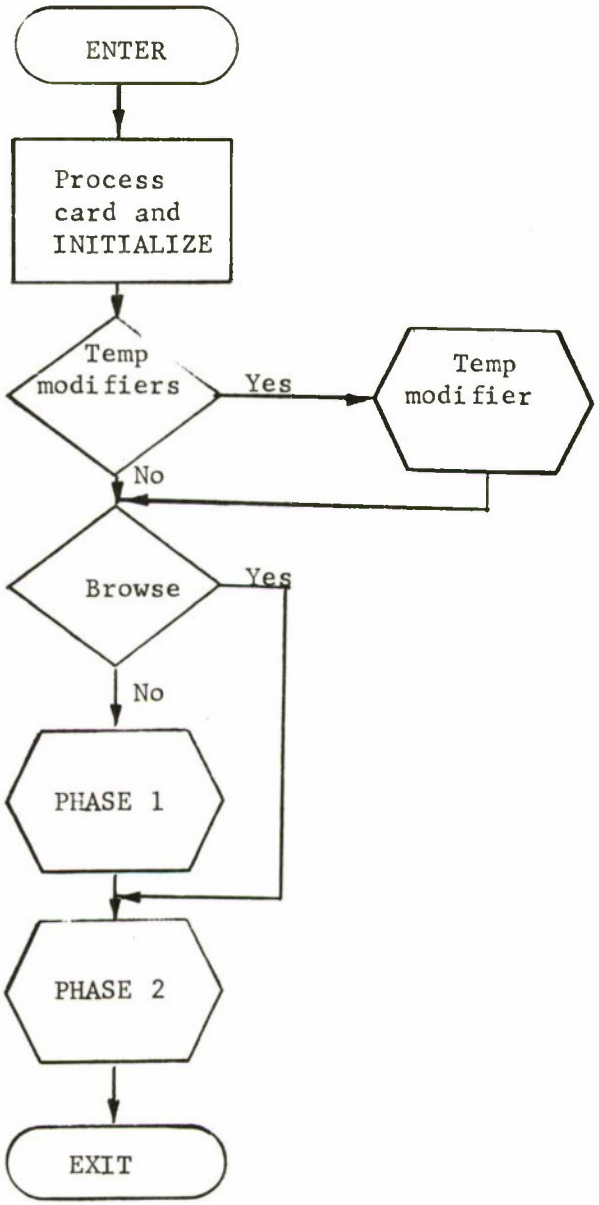
APPENDIX B

PROGRAMS AND SUBPROGRAMS

EXECUTIVE PROGRAM – FMEXEC

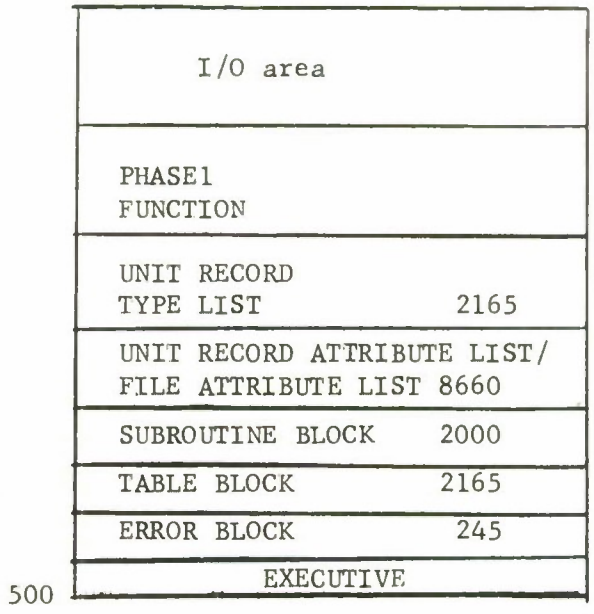
NAME:	File Management Executive
CLASS:	Program
ORIGIN:	00500
FUNCTION:	To determine and oversee the operation of the file update subprograms.
CALLING SEQUENCE:	Call FMEXEC from console
INPUT:	File Management Executive Control Cards and, if desired, logical operator temporary modifiers
OUTPUT:	Execution of file management processes as specified
ERROR RETURN:	Illegal control card directives
ACCESSED SDA's:	Control Set Directory
ACCESSED PROGRAMS:	TEMPMOD PHASE 1 and associated subprograms PHASE 2 and associated subprograms

REMARKS: General Executive Flow



EXECUTIVE SUBPROGRAM – PHASE1

NAME:	PHASE1 Function
CLASS:	Subprogram .(FMEXEC)
ORIGIN:	To be determinedduring implementation
FUNCTION:	To process unit record input data for the generation of logical record transaction data under control of the control set directives
CALLING SEQUENCE:	Called and executed by FMEXEC
INPUT:	Unit Record data
OUTPUT:	Logical Record Transaction data
ERROR INTERFACE:	Optional data error interface with OCC. Mandatory printer data error logging
ACCESSED SDA's:	Control Set Tables Subroutines Error List
REMARKS:	Core Map



EXECUTIVE SUBPROGRAM – PHASE2

NAME: PHASE2 Function

CLASS: Subprogram (FMEXEC)

ORIGIN: To be determined during implementation

FUNCTION: To merge the logical record transaction data with the file data, or using the BROWSE feature, scan and error detect/correct file data

CALLING SEQUENCE: Called and executed by FMEXEC

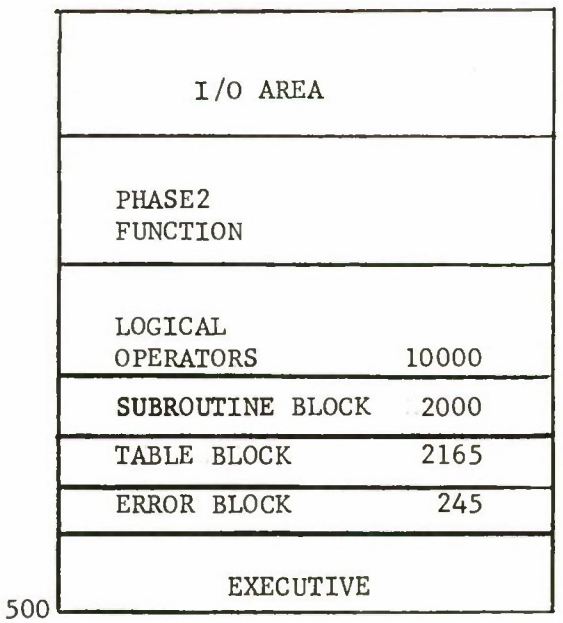
INPUT: Logical record transaction data and/or file data

OUTPUT: Update file data

ERROR INTERFACE: Optional data error interface with OCC. Mandatory printer data error logging.

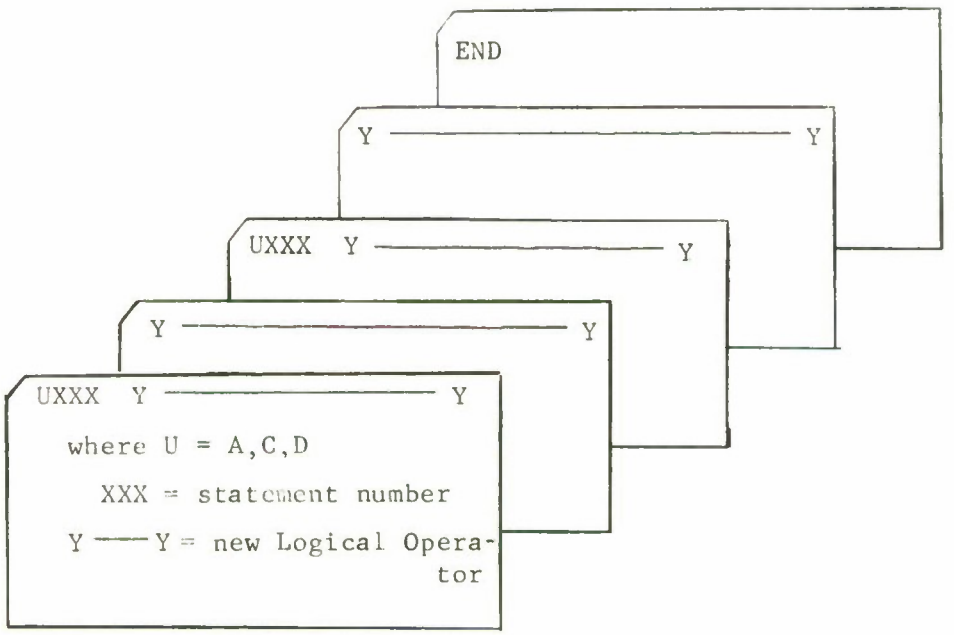
ACCESSED SDA's: Control Set
Tables
Subroutines
Error List

REMARKS: Core Map



EXECUTIVE SUBPROGRAM -- TEMPMOD

NAME:	Temporary Modifier
CLASS:	Subprogram (FMEXEC)
ORIGIN:	To be determined during implementation
FUNCTION:	To modify the logical operators for a given EXECUTIVE pass
CALLING SEQUENCE:	Called and executed by FMEXEC
INPUT:	Logical Operator Update cards followed by an END card



NOTE: Logical operators which continue on more than one card must not have:

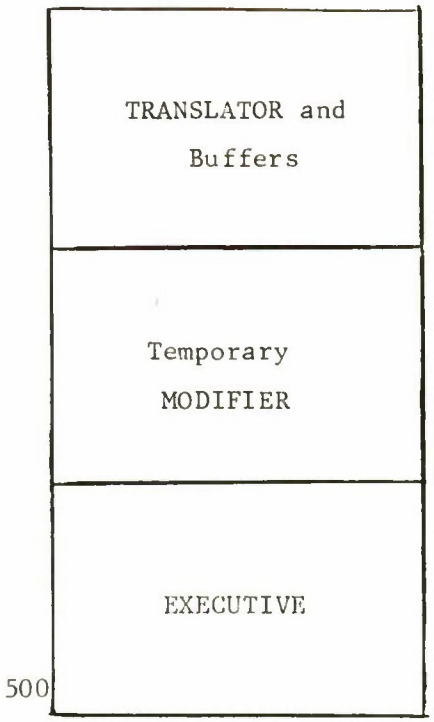
A;
C;
D; or
END

starting in column 1.

OUTPUT: Updated logical operators in compressed form on a disk buffer

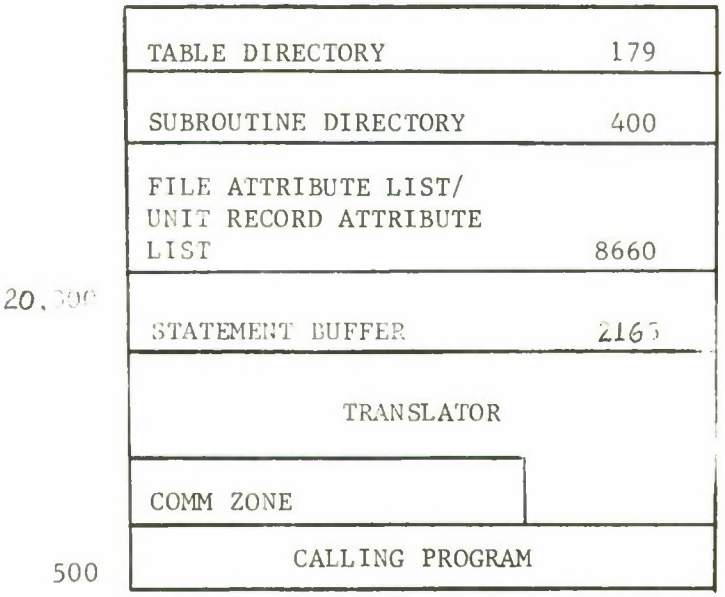
ACCESSED PROGRAMS: TRANS

REMARKS: Core Map



TRANSLATOR SUBPROGRAM -- TRANS

NAME:	Logical Operator Translator
CLASS:	Subprogram (Logical Operator Builder/ Temporary Modifier)
ORIGIN:	To be determined during implementation
FUNCTION:	To translate a raw logical operator into its internal form
CALLING SEQUENCE:	B TRANS in calling program
INPUT:	A complete raw logical operator in the statement buffer
OUTPUT:	Internal representation of the logical operator in the statement buffer
ERROR RETURN:	Error switch set. Error message stored in translator communications zone
ACCESSED SDA's:	Table Directory Subroutine Directory Control Set
REMARKS:	Translator Core Map



Translator Communications Zone contains items such as:

a) SDA of Control Set;
b) File Attribute List location and size;
c) Unit Record Attribute List Location and size;
d) Table Directory size;
e) Subroutine Directory size;
f) Error indicator and message buffer.

UTILITY PROGRAM -- CSDB

NAME: Control Set Directory Builder

CLASS: Program

ORIGIN: 00500

FUNCTION: To maintain the Control Set Directory by creating new control set skeletons, deleting existing control sets, or changing main level information in the Control Set Directory. (May require allocation of new control set SDA and/or reallocating Control Set Directory.)

CALLING SEQUENCE: Call CSDB via console

INPUT: One or more update cards followed by an end card.

Update card format

COLUMN	FIELD	
1	C = Change A = Add D = Delete	
2-11	Set name (left justified)	
12-21	file name (optional)	
22-23	file blocking factor	product ≤ 2800
24-27	file record size	
28-31	merge id location [1,2800]	
32-33	merge id size [1,99]	

COLUMN	FIELD
34	file header on each reel Y = yes N = no b = no headers
35-36	file header length [1,41]
37-77	file header data (left justified)
78-80	CSD

END card format

COLUMN	FIELD
1-3	END

ERRORS:

1. Attempted to delete or change non-existent control set.
2. Attempted to add an existent control set.
3. Control card data in error.

Above errors cause rejection of update card.

1. No end card - requires input.

ACCESSED SDA's:

Control Set Directory
Control Set SDA

UTILITY PROGRAM -- FALB

NAME: File Attribute List Builder

CLASS: Program

ORIGIN: 00500

FUNCTION: To create or overwrite the file attribute list for a given control set and call for retranslation of logical operators, if required.

CALLING SEQUENCE: Call FALB via console

INPUT: A control card and one or more attribute update cards per block, one or more blocks followed by an end card.

Control card format

COLUMN	FIELD
1-11	CONTROLbSET
13-22	Control set name
24	Avoid retranslation of logical operators

b = yes
N = no

Attribute update card format

COLUMN	FIELD
1-10	Attribute name
12-13	Type

A = Alphabetic
AN = Alphanumeric
N = Numeric

Attribute update card format (conc.)

COLUMN	FIELD
15-18	Field size
20-23	Relative location within file record (optional)
78-80	FAL

End card format

COLUMN	FIELD
1-3	END

OUTPUT: File Attribute List built in the Control Set structure.

ERRORS: Invalid control card, Invalid attribute cards } ignore batch

No end card - end card required

ACCESSED SDA's: Control Set Directory
Control Set

ACCESSED PROGRAMS: TEMPMOD

UTILITY PROGRAM -- URALB

NAME: Unit Record Attribute List Builder

CLASS: Program

ORIGIN: 00500

FUNCTION: To create or overwrite the Unit Record Attribute List for a given control set and call for retranslation of the logical operators if required.

CALLING SEQUENCE: Call URALB via console

INPUT: A control card, a Unit Record Data Card, one or more Unit Record identifier cards with associated Unit Record Attribute cards per block, one or more blocks followed by an end card.

Control card format

COLUMN	FIELD
1-11	CONTROLbSET
13-22	Control set name
24	Avoid retranslation of logical operators b = yes N = no

Unit Record data card format

COLUMN	FIELD
1-11	UNITbRECORD
13	Unit record id size [1,5]
13-18	Unit record id location within unit record [1,2800]

Unit Record data card format

COLUMN	FIELD	
20-23	Unit record size	product ≤ 2800
25-26	Unit record blocking factor	

Unit Record Attribute card format

COLUMN	FIELD
6-15	Attribute name
16-17	Type A = Alphabetic AN = Alphanumeric N = Numeric
18-21	Location with unit record
22-25	Field size
26	Conversion T = Table lookup S = Subroutine b = None
27-32	Conversion table or subroutine name
33-73	Conversion parameters
78-80	URA

End card format

COLUMN	FIELD
1-3	END

OUTPUT:	Unit Record Attribute List structure in control set and if redefinition, appeal to Logical Operator retranslation.
ERRORS:	Errors in definition cards cause rejection of update. End card and proper input sequence is mandatory.
ACCESSED SDA's:	Control Set Directory Control Set Subroutine Directory Table Directory
ACCESSED PROGRAMS:	TEMPMOD

UTILITY PROGRAM -- LOB

NAME: Logical Operator Builder

CLASS: Program

ORIGIN: 00500

FUNCTION: To create or retranslate the logical operator for a given control set.

CALLING SEQUENCE: Call LOB via console

INPUT: A control card followed by one or more logical operator data cards per block, one or more blocks followed by an end card.

Control card format

COLUMN	FIELD
1-11	CONTROLbSET
13-22	Control set name

Logical Operator data and format

Free format card stream of Logical Operators.

No card may contain either

CONTROLbSET

or

END

starting in column 1.

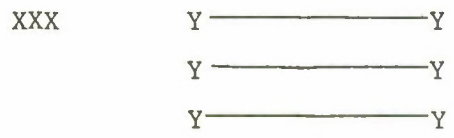
End card format

COLUMN	FIELD
1-3	END

OUTPUT:

Translated Logical Operators stored on disk. Raw Logical Operators stored on scratch tape.

Logical Operator statement number list on printer.



where XXX is the assigned statement number

YY———Y is raw statement.

ERRORS:

Illegal control card causes rejection of data cards.

Logical Operator format errors

End card is mandatory.

ACCESSED SDA's:

Control Set Directory
Control Set
Table Directory
Subroutine Directory

ACCESSED PROGRAMS:

TRANS

UTILITY PROGRAM -- SUBDB

NAME: Subroutine Directory Builder

CLASS: Program

ORIGIN: 00500

FUNCTION: To maintain the subroutine directory by adding new entries, deleting old entries and changing SDA's for existing subroutine names. Reallocation of Directory SDA may be necessary.

CALLING SEQUENCE: Call SUBDB via console

INPUT: One or more update cards followed by an END card.

Update card format

COLUMN	FIELD
1	A = Add D = Delete C = Change
3-8	Subroutine name (left adjusted)
10-13	SDA of subroutine (SDA must be in Mode 5 1-5 relative records long)

End card format

COLUMN	FIELD
1-3	END

OUTPUT: Updated Subroutine Directory entries

ERRORS:

1. Attempted to delete or change a non-existent table entry;

2. Attempted to add an entry with name already defined;

3. Bad card format;

4. Illegal SDA type.

Above errors cause rejection of the erroneous card.

5. No end card. End card necessary to terminate run.

ACCESSED SDA's:

SDAT

Subroutine Directory (implementor will allocate and assign an SDA for this directory and request 1 record of Mode 5 storage).

UTILITY PROGRAM -- TABB

NAME: Table Builder

CLASS: Program

ORIGIN: 00500

FUNCTION: To add new tables, change existing tables and delete existing tables (may involve reallocation of Table Directory and Table SDA).

CALLING SEQUENCE: Call TABB via console.

INPUT: One control card and a number of table value cards per block, one or more blocks followed by an end card.

Control card format

COLUMN	FIELD
1	C = Replace table D = Delete table A = Add table
3-8	Table name (left adjusted)
9-10	Lookup argument size [1,80]
12-13	Value size [0,79]
	NOTE: Argument size X value size ≤ 80.
15	C = Conversion table L = Legal value table (no values)
78-80	TAB

Table data column

COLUMN	FIELD
1-n	Argument (free format) where n is argument size
(n+1)-80	Value (free format)

End Card

COLUMN	FIELD
1-3	END

OUTPUT: Updated entries in the Table Directory and TABLE SDA.

ERRORS:

1. Attempted to delete or change non-existent table;
2. Attempted to add table already defined;
3. Error in control cards.

Above cause the bypassing of all cards for the erroneous block.

1. No end card; requires end card.

ACCESSED SDA's: Table Directory (implementor will initially assign one record of Mode 7).

Table SDA (implementor will initially assign one record of Mode 3).

UTILITY PROGRAM -- ERRDB

NAME: Error Directory Builder

CLASS: Program

ORIGIN: 00500

FUNCTION: To update an entry in the error directory by replacing the contents of the message with card data.

CALLING SEQUENCE: Call ERRDB via console.

INPUT: One or more update cards followed by an end card.

Update card format

COLUMN	FIELD
1-3	Error number [1,200]
4-80	Error text - free format

End card format

COLUMN	FIELD
1-3	END

OUTPUT: Updated entires in the Error Directory.

ERROR: Illegal error number--ignore card.
No end card--requires end card.

ACCESSED SDA's: Error Directory (implementor will allocate an SDA of Mode 8, 67 records. All records will be initially cleared).

UTILITY PROGRAM -- UPRINT

NAME: Utility Print

CLASS: Program

ORIGIN: 00500

FUNCTION: To list in appropriate formats

1. Control Set Directory.
2. Control Set File Attributes and Unit Record Structure and Attributes.
3. Table data.
4. Subroutine Directory.

CALLING SEQUENCE: Enter via the console the following:

1. UPRINTbSETbDIR

   The above lists all entries in the Control Set Directory.

2. UPRINTbSETbXX...X where XX...X is a control set identifier

   The above lists the File Attributes and Unit Record Structure and Attributes for the given control set.

3. UPRINTbSETbALL

   The above performs the function of step (2) for all control sets.

4. UPRINTbSUB

   The above lists the Subroutine Directory.

5. UPRINTbTAB

   The above lists the Table Directory and also lists the data in the Tables.

6. UPRINTbERR

   The above lists the non-blank entries in the error table.

INPUT: Appropriate system SDA's.

OUTPUT: The format of the printing will be as follows relating formats number with option number in CALLING SEQUENCE.

1. CONTROL SETS DATE PAGE

SET NAME	FILE NAME	ETC.
X———X	X———X	
X———X	X———X	
X———X	X———X	

2&3. CONTROL SET X———X CONTENTS DATE PAGE

FILE ATTRIBUTES

ATTRIBUTE NAME	TYPE	ETC.
X———X	X—X	
X———X	X—X	
.	.	
.	.	
.	.	
X———X	X—X	

UNIT RECORD DATA

BLOCKING FACTOR	RECORD LENGTH	TYPE ID SIZE	ETC.
X———X	X——X	X	

RECORD TYPE = X———X

ATTRIBUTE NAME	TYPE	ETC.
X———X	X—X	
X———X	X—X	

ATTRIBUTE
NAME TYPE ETC.

. .
. .
. .

X————X X—X

RECORD TYPE = X————X

ETC.

4. SUBROUTINES DATE PAGE

NAME SDA

X———X X———X
X———X X———X

. .
. .
. .

X———X X———X

5. TABLES DATE PAGE

ARG
NAME TYPE LENGTH ETC.

X—X X—X X———X

ARG VALUE

X——X X——X
X——X X——X
. .
. .
. .
X——X X——X

NAME	TYPE	ARG LENGTH	ETC.
X—X	X—X	X———X	

ARG	VALUE
X——X	X——X

ETC.

6. ERROR LIST DATE PAGE

NUMBER	TEXT
X——X	X————X
X——X	X————X

ETC.

ACCESSED SDA's: Appropriate System SDA's

OCC INTERFACE ROUTINE -- OCCIF

NAME:	OCC Interface (OCCIF)
CLASS:	OCC Program
ORIGIN:	To be determined during implementation
FUNCTION:	To receive an error stream from the 1410; format the OCC screen with the error; accept user corrections, if desired, and transmit the corrected data to the 1410.
CALLING SEQUENCE:	Internal 1410 write out to BR-90
INPUT:	Stream from 1410
	M....MV....VE....E
	where M....M is the merge id V....V is the error value E....E is the error message text
OUTPUT:	Stream to 1410
	XV....V
	where X is the status (pad, retry with correction, or ignore)
	V....V is new value

APPENDIX C

TABLE FORMATS

NAME: Update Control Set Directory

PURPOSE: Initial link to the Control Set

SIZE: Variable, one entry per Control Set, five entries on each record, mode L5.

AREA FORMAT: a)

A	B	C	D	E
1	81	161	241	321

b)

			v b	A	B
1	81	161	241	321	

Entries A, B, ···E are identical in format per disk record.
Format a describes disk record with five entries
Format b describes disk record with two entries.

Entry Format: Size: Fixed length, 80 characters.

Description:

v FFFFFFFFFF	G	HH	III.....I	KK	LLLL	MMMM	NN	SSSS	XXXXXXXXXX
1	11	12	14	55	57	61	65	67	71

FFFF...F File Name (optional)

G Y, file header on each reel
N, file header not on each reel

HH File Header Length

II.....I First 41 characters of File Header

KK Tape File Blocking Factor

LLLL	Tape File Record Length
MMMM	Merge ID Location (within file record)
NN	Merge ID Length
SSSS	SDA of Control Set
XXX....X	Control Set Name

Remark: In general, all alphanumeric items will be left adjusted and padded with blanks; all numeric items will be right adjusted and padded with zeroes.

NAME: Update Control Set

PURPOSE: Provides overall control information for all serial file generation, update and browse programs. Each Update Control Set contains:

a) the file description attribute list

b) the unit input record(s) attribute list(s)

c) the compressed form of the logical operators.

LENGTH: Variable, mode L1, maximum length of one control set is 85 relative records.

AREA FORMAT:

Pointers to lists and other control info	Rel. Record ØØØØ
File Attribute List (Up to 20 relative records)	Rel. Record ØØØ1
Unit Record Pointer List (1 rel. record)	
Unit Record Attribute List (Up to 56 relative records)	
Logical Operators (compressed) (Up to 5 relative records)	

DETAIL AREA FORMAT: Update Control Set Pointer Record

FORMAT:

v AAAA	v BBBB	v CCCC	v DDDD	v EEEE	v F	v GGGG	v HHHH	v II	v KKKK	v LLLL
1	5	9	1 3	1 7	2 1	2 2	2 6	3 0	3 2	3 6

v M	v NNNN	v OO		
4 0	4 1	4 5	4 7	2 1 6 5

AAAA = SDA

BBBB = Number of relative records this SDA

CCCC = Starting relative record of File Attribute List

DDDD = Number of records in File Attribute List

EEEE = Starting relative record of Unit Record Pointer List

F = Number of relative records in Unit Record Pointer List (always 1)

GGGG = Starting relative record of Unit Record Attribute List

HHHH = Number of relative recordsin Unit Record Attribute List

II = Unit Record Blocking Factor

KKKK = Unit Record Size

LLLL = Relative start location (in Unit Record) of Unit Record Type ID.

M = Length of Unit Record Type ID.

NNNN = Starting relative record of Logical Operators.

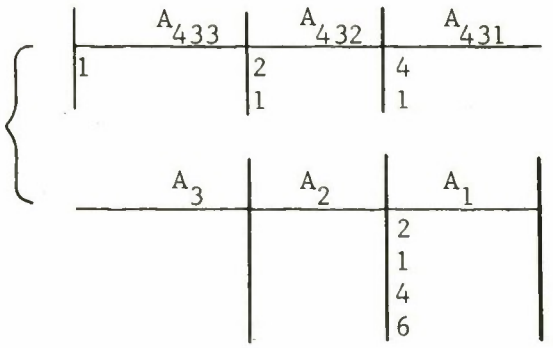
OO = Number of relative records of Logical Operators.

DETAIL AREA FORMAT: File Attribute List

LENGTH: Variable, maximum of 20 relative records, packed in blocks of 4 records from right to left for table lookup. A maximum of 2000 unique attributes may be stored per file as 20 character words.

FORMAT:

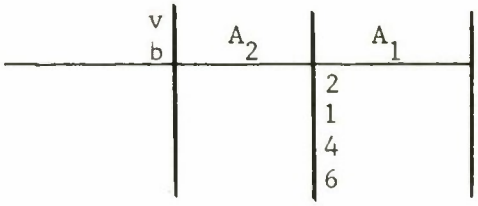
a)



1st record of block

4th relative record of block

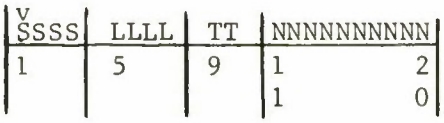
b)



4th relative record of block

Format a illustrates a completely filled block. Format b illustrates a partially filled block with a short field set to stop the table search. The contents of the first three relative records are irrelevant.

Entry Format:



SSSS = Field Size
LLLL = Field Relative Location (File Record)
TT = Data Type
NN...N = Attribute Name

DETAILED AREA FORMAT: Unit Record Pointer List

LENGTH: One relative record, each entry is 18 characters long. Up to 100 entries are permitted.

FORMAT:

	v b	A_3	A_2	A_1
	2 1 1 1	2 1 1 2	2 1 3 0	2 1 4 8 2 1 6 5

The entries are packed from right to left, the leftmost entry is preceded by a wordmark-blank to end a table search.

Entry Format:

v NNNN	LLLL	RRRR	C	XXXXX
1	5	9	1 3	1 1 4 8

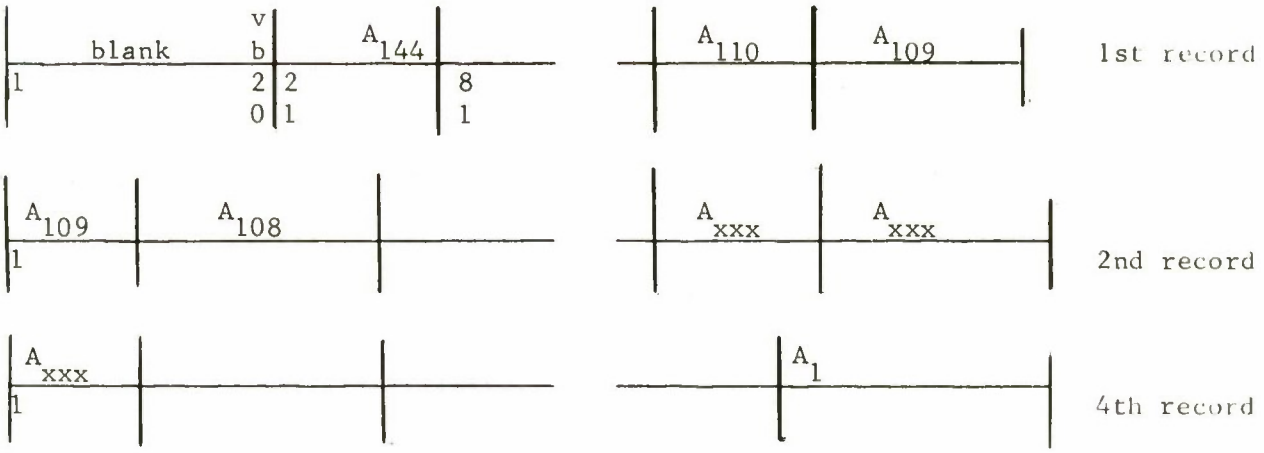
NNNN = Number of attributes for this type record
LLLL = Length of attribute list for this type record
RRRR = Relative entry in Unit Attribute List stream
C = Sequence Control (C, F, N, I)
XXXXX = Unit Record ID Value.

DETAILED AREA FORMAT: Unit Record Attribute List

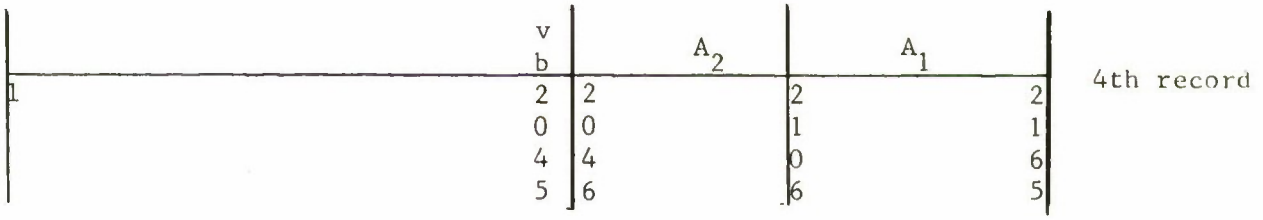
LENGTH: Variable; up to 56 relative records. Entries are 60 characters long and are packed in blocks of 4 records from right to left.

FORMAT:

a)



b)



Format a depicts block of 4 records with maximal number of entries.
Format b shows a block with only 2 entries. The contents of records 1-3 in this block are irrelevant.

Entry Format:

v	TTT...T	C	DD	LLLL	SSSS	RRRR	NNNNNNNNNN
1	3	3	3	3	4	4	5
	5	6	7	9	3	7	1

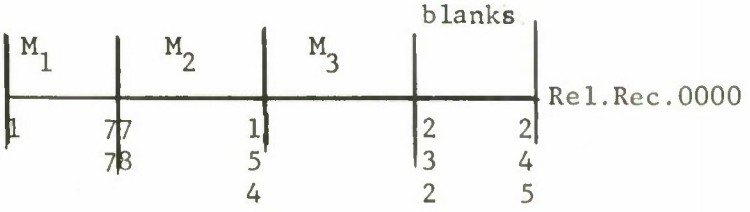
TTT···T	=	Table or Routine Name and Parameters
C	=	Conversion Indicator
DD	=	Data Type (A, N, AN)
LLLL	=	Relative location within file record
SSSS	=	Field Length
RRRR	=	Relative location within unit record
NN.....N	=	Attribute Name

DETAILED AREA FORMAT:	Logical Operators Table
LENGTH:	Variable, maximum of 5 relative records. The logical operators may have a maximum length of 10000 characters.
FORMAT:	The logical operators are stored in a continuous stream from left to right and may overflow from one relative record to the next relative record.

NAME: Error Message Table

LENGTH: Fixed, Mode M10, 67 relative records. A maximum of 200 error messages may be stored in the table. Each message has a maximum length of 77 characters.

FORMAT:



Relative records 1 through 67 are identical in format to relative record 0.

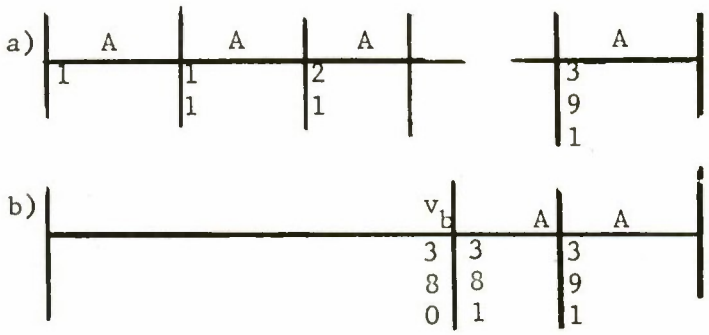
Entry Format: Each error message is left-adjusted in its 77 character slot.

NAME: Subroutine Directory

PURPOSE: Provides linkage to the user subroutines.

SIZE: Variable, mode L5, one entry per subroutine, up to 40 entries each record

AREA FORMAT:



Format a describes disk record with 40 entries.
Format b describes disk record with 2 entries.

Entry Format:

Size: Fixed length, 10 characters.

Description:

XXXXXX = Subroutine Name

SSSS = Corresponding SDA

Remark: Subroutine Name must be at least two characters long. The length of a subroutine is limited to 2000 characters. All subroutines will be assembled with the SAP assembler and stored in up to five relative records of Mode L5.

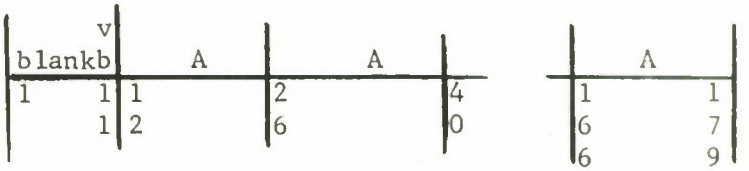
NAME: Lookup Table Directory

PURPOSE: Provides linkage to user lookup tables

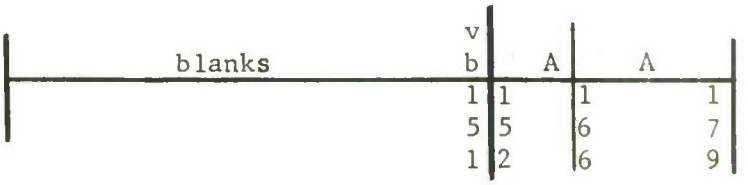
SIZE: Variable, mode L1Ø, one entry per lookup table, up to 12 entries each record.

AREA FORMAT:

a)



b)



Format a describes disk record with 12 entries.
Format b describes disk record with 2 entries.

Entry Format:

Size: Fixed length, 14 characters

Description:

XXXXXX = Table Name

RRRR = Relative Record

NNNN = Number of Relative Records

Remarks: Table Name must be at least two characters long. Set A bit over low order character in RRRR if lookup argument in corresponding table is one character.

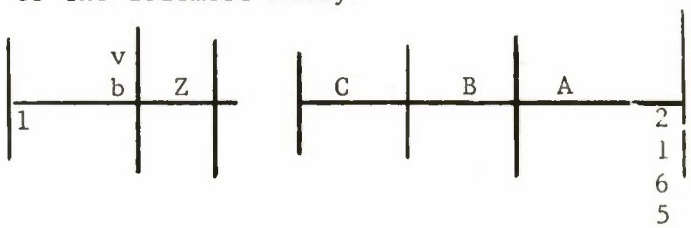
Set A bit over tens position of RRRR to indicate a legal value table (no A bit indicates conversion table).

NAME: Lookup Table Set

PURPOSE: Contains (user supplied) Lookup Tables

SIZE: Variable, Mode L1, up to 27 entries per relative record.

AREA FORMAT: All entries are packed from right to left into the relative record. A $\overset{v}{b}$ is placed to the leftmost entry.



Entry Format: Size: Variable length, maximum length is 80 characters

Description: Wordmark placed over high order character, the sum of the sizes of the table argument and the table value must be not more than 80 characters.

VVV...V = Table Value

AAA...A = Table Argument

APPENDIX D

EXECUTIVE CONTROL CARD FORMATS

The EXECUTIVE requires two control cards with the following formats:

CONTROL CARD 1

COLUMN	FIELD
1 - 4	CTL1
6 - 13	Type of run UPDATEbb=update run GENERATE=generate run BROWSEbb=browse run
15 - 24	Control set name, left-justified
26 - 29	Unit record input source CARD=card input TAPE=tape input
31 - 38	Temporary modification of logical operators TEMPbMOD=yes bbbbbbbb=no
40 - 46	Tape I/O overlap option OVERLAP=yes bbbbbbb=no
48 - 54	Input update type MIXEDbb=mixed option (implies use of $$UPDATE* unit record attribute) ALL/CHb=all updates are changes ALL/ADD=all updates are adds ALL/DEL=all updates are deletes ADD/CHb=all updates are either adds or changes

*If the mixed mode (A,C, or D) of update is selected, one, and only one, $$UPDATE unit record attribute must be defined for a control set. The user need only provide the relative position within the unit record. Implied definitions for this field are as follows:

SIZE = 1 character;
TYPE = alphabetic (A)
LEGAL VALUES = A,C, or D for add, change, or delete.

COLUMN	FIELD
56 - 58	Sort sequence of file and input records (by merge id in 1410 collating sequence) ASC=ascending order DSC=descending order
60 - 66	Mode of operation ONLINEb=OCC error interface OFFLINE=no OCC error interface

CONTROL CARD 2

COLUMN	FIELD
1 - 4	CTL2
6	Error padding character (any legal 1410 character)
8	Allow input unit record sequence errors and/or record repetitions Y=yes, ignore errors b=no, log errors to printer
10	Do not generate logical update transaction record if $$COUNT* is illegal Y=yes, do not generate record b=no, generate record
12 - 15, 17 - 20	Size and position of BREAK** field in file record. Blank entries imply no break option.

---

* If the user wishes to perform a unit record count check on unit record input, one, and only one, $$COUNT unit record attribute must be defined for a control set. The user supplies the relative position of the count field and the size of the count field (field size is a maximum of three positions). The TYPE is assumed as numeric.

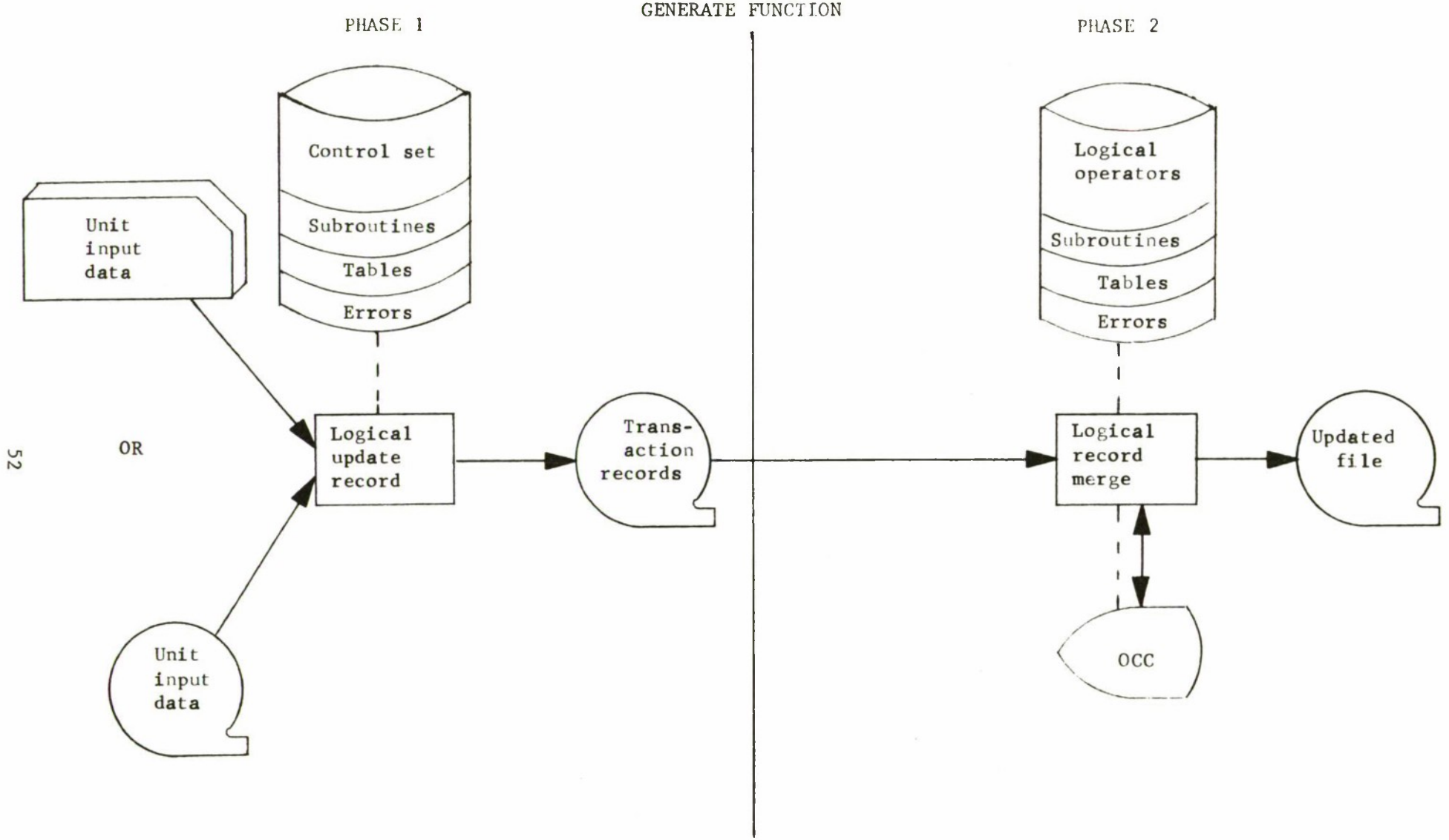
** BREAK OPTION -- If desired, the user may define a field on the file record which when the value changes from one record to the next will start writing records on a new tape.

COLUMN	FIELD
22 - 23	Size of tape file trailer records [01,41]. Blank entry indicates no trailer records
25 - 66	Trailer record data (left-justified) if field (22 - 23) is non-blank.

APPENDIX E

SYSTEM DATA FLOW AND CONTROL

GENERATE FUNCTION

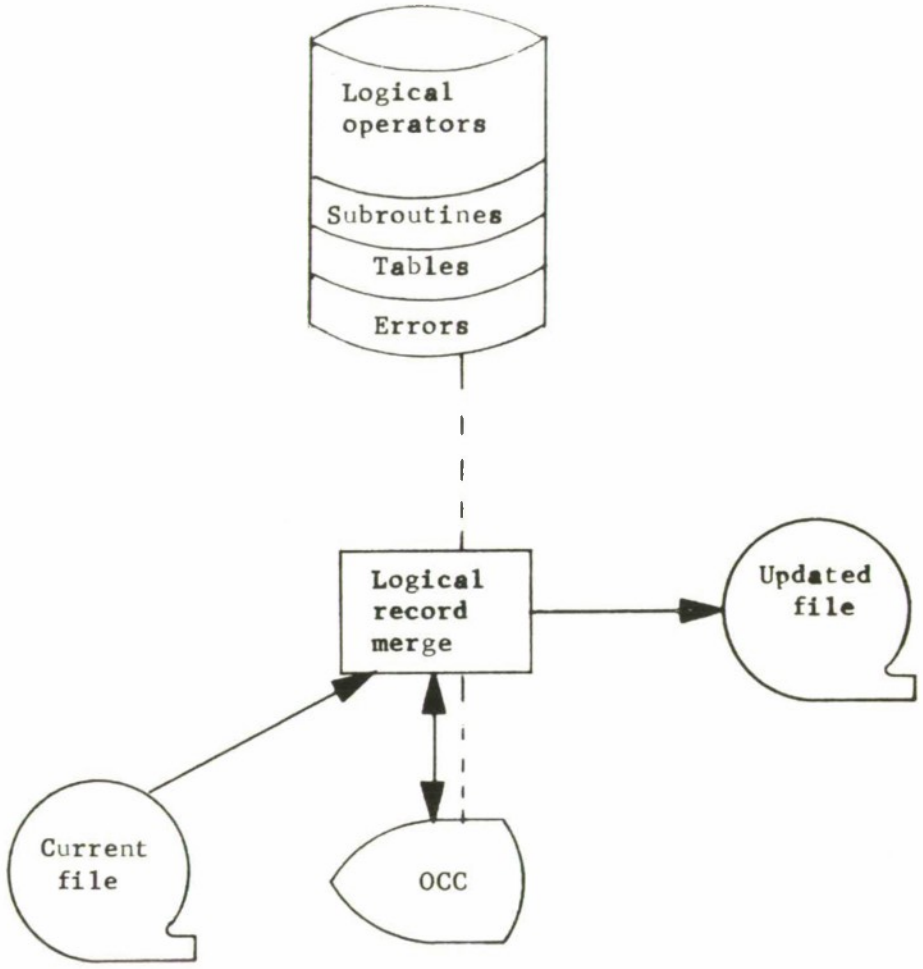


SYSTEM DATA FLOW AND CONTROL

BROWSE FUNCTION

PHASE 1

PHASE 2



Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
The MITRE Corporation Bedford, Massachusetts	UNCLASSIFIED
	2b. GROUP N/A

3. REPORT TITLE

DATA FLOW IMPROVEMENTS: DESIGN SPECIFICATIONS FOR AFICCS SERIAL FILE MANIPULATORS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

N/A

5. AUTHOR(S) (First name, middle initial, last name)

O. Beebe, J. Penney, J. Terrasi

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
November 1968	58	--

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
AF19(628)-5165	
b. PROJECT NO. 512V	ESD-TR-68-290
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) MTR-711
d.	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
N/A	Directorate of Planning and Technology, Electronic Systems Division, Air Force Systems Command, L. G. Hanscom Field, Bedford, Massachusetts

13. ABSTRACT

This document defines the overall design for an AFICCS generalized serial tape file management capability. This capability provides three management functions - generate, update, and browse.

DD FORM 1 NOV 65 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
AFICCS						
1410 FILE MAINTENANCE						
SERIAL FILE						
UPDATE						
GENERATE						